# CMPT 431: Distributed Systems

## Project, Spring 2019

This project is to be done in groups of 4 students!

In this project, your group will design a real-time multiplayer game that we shall call **Deny and Conquer**.

**Gameplay**

The game board is a square divided into boxes of equal size. The number of boxes shall be configurable. The game is played by four players, each having a pen of different colour. The thickness of the pen is the same for all players, and is also configurable. The objective is to deny your opponents filling the most number of boxes, by taking over as many boxes as you can. To take over a box, you must colour the box with your pen, and at least *X*% of the area of the box must be coloured to be considered taken over by you. Otherwise, another player can try taking over the box. *X* is also configurable. Once a box has been taken over by a given player; i.e., that player has coloured at least *X*% of the area of a box, then the box turns entirely to the colour of the player and can no longer be taken over by any other player. At the end of the game; i.e., when all boxes have been taken over, whoever has the most number of boxes will win the game. Having more than one winner is possible. An example is shown in Figure 1.

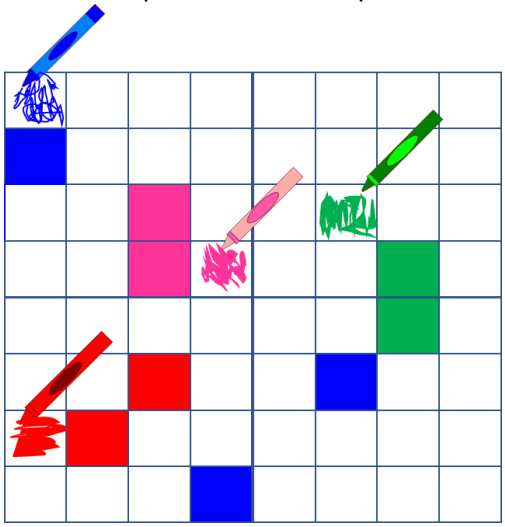

Figure 1. An example gameplay between 4 players in Deny and Conquer.

**Game Mechanics:**

To take over a white box, a player must click anywhere inside that box and hold the click button down, while scribbling inside the box. Once the player lets go of the click button, the game should calculate the %area that is coloured. If it is more than *X%* of the area of that box, then the player has taken over that box, and the colour of the box shall change completely to the colour of the player. No other player can take over that box anymore. But, if it is less than *X%*, the player has not taken over the box, in which case the game immediately changes the box back to pure white, and the box is up for grabs again!

While a player is scribbling in a box, that box is no longer available to other players. If those other players click in that box, they should not be able to draw anything in it.

**Requirements:**

The game shall be a client-server based distributed system supporting concurrency, coordination, and fault tolerance by replication.

The server is run by the player who starts the game session. All players (including the player who started the session) connect to that server as clients.

Concurrency requires that once a player clicks in a box and while scribbling, that box should be "locked" and made unavailable to other players, until the first player lets go of the click button.

Obviously the player who is running the sever is at an advantage: his/her clicks reach the server faster than others. To avoid that, coordination requires that the clocks of all players be synchronized, and the server to check the timestamp of each player's message. In case more than one player has clicked in a box at almost the same time, whoever has the smaller timestamp should be given access to that box.

It is not unusual for the computer of the player who is running the server to go down. This could be because of network problems, that player's computer crashing, or the player getting mad that s/he is losing and just stopping the server. In such a case, the system should choose another player's computer to run the server for the remaining players, starting from the last known state of the game. This requires fault tolerance by replication.

Obviously there will be a pause while the other server is coming up. During this pause, the game should display some message to the remaining players so they know something is wrong and they need to wait. This message can be a rotating circle (like the buffering sign we get from YouTube) or any other similar visual message.

**Technical Rules:**

- You can use any programming language that you like.
- For the frontend, you can use any existing graphics or GUI library or framework. Make your life easy for the frontend as much as possible. Don't overdo the GUI. A simple and functional GUI is enough.
- For the backend (client and server system), you cannot use any existing gaming, client-server, messaging, remote calling, or other distributed system middleware or frameworks. Everything must be written from scratch. You must use networking sockets (TCP or UDP) directly.

**Deliverables:**

1. A working demo, to be presented in the last 4 classes, as shown in the course Syllabus.
2. A project report, due April 12 in Canvas, describing:
   - The architecture
   - All design choices: why did you choose this design instead of an alternative design? Why TCP instead of UDP, why …?
   - Commented source code of the client and the server

**Marking Scheme:**

Working Demo 45%
Project report 55%